The Visitor Pattern

Gerald Britton

Pluralsight Author

@GeraldBritton www.linkedin.com/in/geraldbritton

Overview

Classification: Behavioral

Add new abilities to an object structure

Build abstractions for new functionality

Keep the new capabilities separate

Reduce cost and risk

Can break encapsulation

Demo

Motivating Example:

- Family tree from Composite Pattern
- Add pretty print feature
- Just add some more code!

Visitor Pattern

Example:

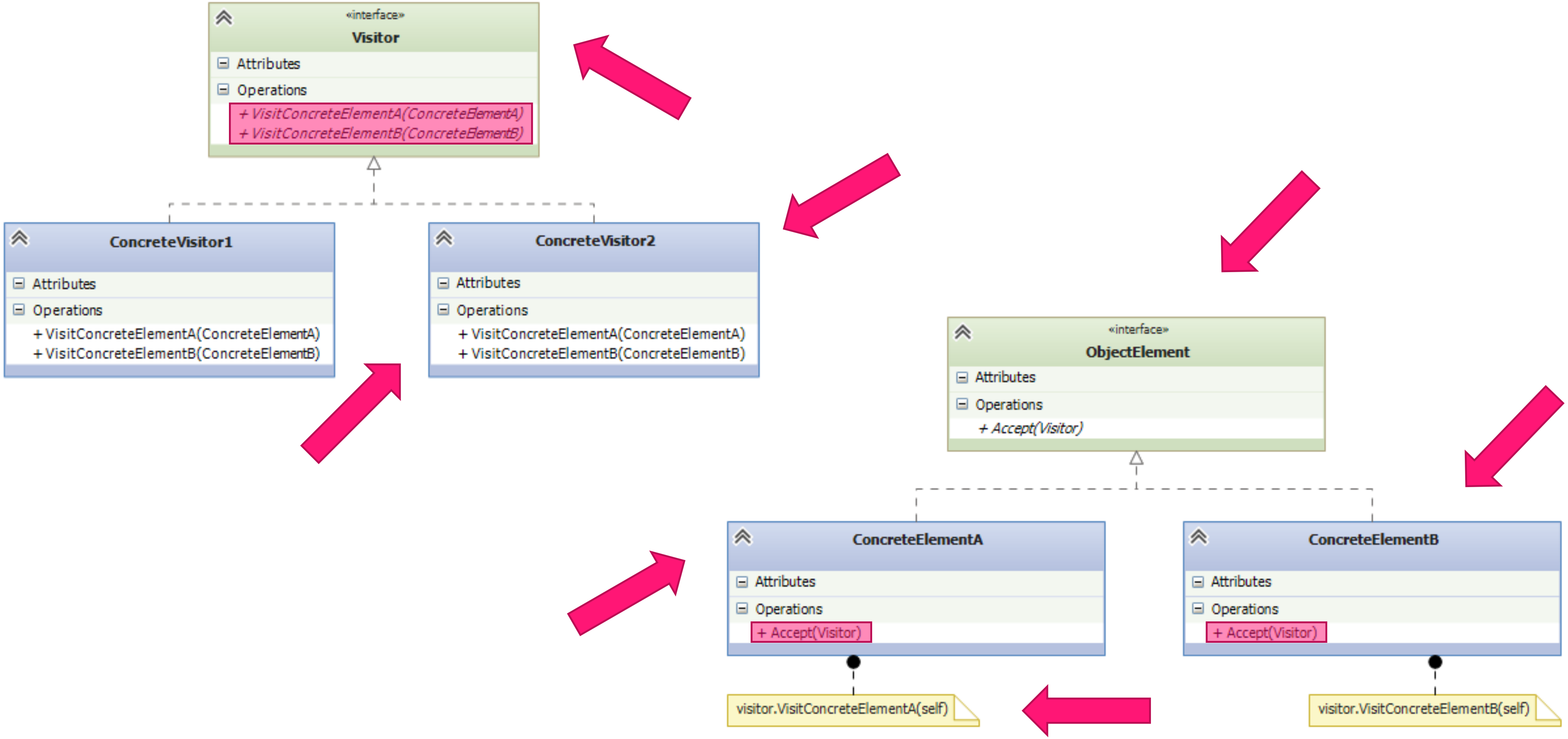
- Visit your house, a library or museum
- Visitor brings a camera or bag
- Take pictures or shop at store

Visitor visits an object

- Gets access to the object’s contents
- Breaks encapsulation

Implements desired functionality

Visitor Pattern Structure


«interface»
Visitor
Attributes
Operations
+ VisitConcreteElementA(ConcreteElementA)
+ VisitConcreteElementB(ConcreteElementB)
ConcreteVisitor1
Attributes
Operations
+ VisitConcreteElementA(ConcreteElementA)
+ VisitConcreteElementB(ConcreteElementB)
ConcreteVisitor2
Attributes
Operations
+ VisitConcreteElementA(ConcreteElementA)
+ VisitConcreteElementB(ConcreteElementB)
«interface»
ObjectElement
Attributes
Operations
+ Accept(Visitor)
ConcreteElementA
Attributes
Operations
+ Accept(Visitor)
ConcreteElementB
Attributes
Operations
+ Accept(Visitor)
visitor.VisitConcreteElementA(self)
visitor.VisitConcreteElementB(self)

Demo

Implement the Visitor Pattern

Put pretty print logic in the Visitor

Demo

Find oldest person

Use Visitor pattern

Consequences

Easy to add new applications

Harder to change the data model

Works across class hierarchies

Accumulate state

Breaks encapsulation

In Python, class decorators can replace Visitors

Summary

Object structure with many classes

- Separate new functionality

Many operations to perform

- Avoids polluting the object structure

Data model classes rarely change

- No ongoing changes to the Visitors

Alternative: Python class decorator